2017 IEEE International Conference on Technological Advancements in Power and Energy (TAP Energy)

BusTimer: An Android Based Application for Generating Bus Schedules Using Crowdsourcing

Sandheep S[1], Harry John[2], Harikumar A[3], Vinitha Panicker J[4]
Department of Computer Science and Engineering,
Amrita Vishwa Vidyapeetham, Amritapuri, India.
[1]sandheep.sreekumar@gmail.com, [2]harry671003@gmail.com, [3]ihari1993@gmail.com, [4]vinithapanicker@gmail.com

***Abstract*— The fast growth of Indian population has triggered a greater need for well-organized public transport service. Surveys show that approximately 53 million persons travel through almost 400,000 buses twice daily. Often women are seen taking private transportation at night which really compromises their safety. We did some survey and found that most of the women prefer to take public bus transport during night time, but they are forced to take private transport due to lack of an easy means to check the bus schedules. In this paper, we describe and evaluate an android based application that uses crowdsourcing technique for generating bus schedules, even for short distance travel. The system uses the android accelerometer for detecting the bus stops. Then GPS is activated and the coordinates of the stop are sent to the server. The server runs a scheduling algorithm and updates the schedule. And whenever a user of the application enters the source and destination, the corresponding bus schedule will be displayed. The system is implemented and tested using the crowdsourced data populated by twenty five volunteers, as they travelled by public bus transport and the results proved the superiority of the system.**

Keywords— android application; accelerometer; bus schedule; crowdsourcing

I. INTRODUCTION

Transportation demand in most of the Indian cities has increased significantly, as a result of the natural increase in population as well as due to migration from smaller towns and rural areas. As per surveys, the growth of population for the metropolitan areas such as Mumbai, Kolkata, Delhi, etc. exceeded 10 million residents each. Chennai, Hyderabad, Ahmedabad, and Bangalore each have more than 5 million residents. And 35 metropolitan areas have populations exceeding 1 million, which is almost twice as many as in 1991. Since large cities are far more dependent on public transport than small cities, the need for public transport services has increased faster than overall population growth.

This exponential growth of population results in a high demand on transportation facility and as a result, the number of private vehicles goes on increasing which in turn results in traffic congestion, air pollution, environmental degradation, higher road accidents etc. The only way to reduce the traffic problem is to make the people use the public transportation facility. Among various means of public transportation available, the bus transport is preferable because of its door-to-door accessibility and flexibility in operation. Although the demand for bus transportation is high, lack of availability of bus schedules is one of the key factors which makes most of the people to think the other side.

This paper proposes a cost effective method of generating bus schedules by tracking a user's position while he/she is traveling by bus. The whole system allows tracking the user's mobility using a mobile phone which is equipped with an internal GPS receiver, a GPRS transmitter, and an accelerometer. A mobile phone application, BusTimer, has been developed and deployed on an android mobile phone whose functionality is to track accelerometer readings and identify the bus stops. Whenever a stop is identified, the GPS location and time is obtained and sent it to a remote location by creating a GPRS packet. The data collected from the phone is sent to the server hosted on AWS (Amazon Web Services), which then runs the scheduling algorithm on the data, and updates the bus schedules stored in DynamoDB. All the communications with the server are via HTTP requests and responses are in JSON format. The cost of the BusTimer is low as compared to other systems as we are using inbuilt GPS receiver in a mobile phone instead of a separate handheld GPS device. And the battery drain that may occur due to the usage of GPS is minimized with the help of the accelerometer.

The rest of the paper is organized as follows. Section II provides the various related research work in this area. The system architecture for BusTimer is discussed in section III. Section IV provides the bus schedule generation algorithm. The implementation and the testing details are provided in section V.

II. RELATED WORKS

Android applications are now inevitable in our day today life and new researches are happening on different fields like [1-2]. Crowdsourcing is another key research area in the present scenario. A lot of research work is happening which takes the advantage of crowdsourcing [3-4]. Moreover, the main part of our proposed work is to track the crowdsourcing users so that the bus stops and timings can be collected. Various researches have been done in the past on the different methods of object tracking using GPS technology. Most of the related works mentioned below could be used for tracking an object, whether it is a vehicle, a person or anything [5-9].

978-1-5386-4021-0/17/$31.00 ©2017IEEE

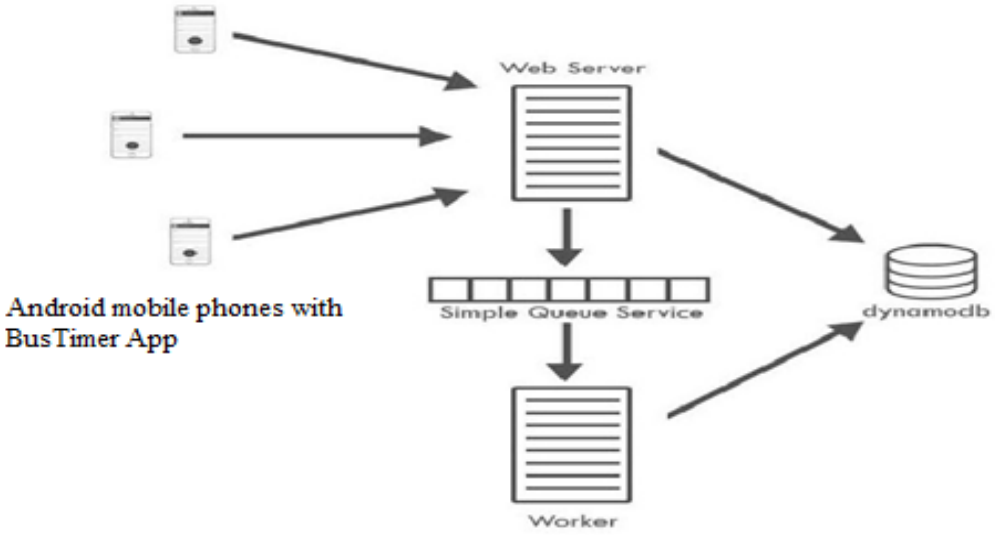


Fig.1 BusTimer System Architecture

.

Hasan K.S.et.al. [5] proposed a cost effective method of object tracking using GPS and GPRS technology.

Fengli Zhang et. al. [6] presented an overview on the fetching and integration of the moving object data stored in a database. It is pre-assumed that the MOD (Mobile Object Data) is fetched and stored from the object through a GPS enabled mobile object.

G. Yuan et.al. [7] proposed a GIS-based vehicle tracking system which uses the GPS data obtained by a GPS device placed within each vehicle and it is transmitted through GPRS. [8] and [9] also uses a similar technology using GIS technology.

Almost all of the applications discussed so far use a handheld GPS receiver device for tracking the location, but we have designed a cost-effective system by using the mobile phone which has an inbuilt GPS receiver. And further the battery drain that may occur by using the GPS is also minimised by using the accelerometer here. GPS is activated only when the accelerometer readings suggest that the bus has reached a stop instead of turning on the GPS all the time. Thus we are suggesting an energy-efficient system as well.

III. BUSTIMER:- SYSTEM ARCHITECTURE

BusTimer application architecture mainly has:

- An android based mobile application front-end.
- A back-end hosted on Amazon Web Service (AWS)

Fig. 1 shows the proposed system architecture.

A. Front-End:- Android based mobile application

BusTimer is an android application that provides the user with accurate bus schedules between any two places. The user interacts with the system through the app. When a user installs the app and runs it for the first time, the app sends a request to the server to register the device and the server responds with a unique user ID (UID). This UID is sent along with all data sent by that user to identify them.

The application has two main functions

1) *Data Collection through Crowdsourcing*: Our system will populate the bus schedules with the help of those registered users who help for crowdsourcing. There is an option available in the application, “Start Crowdsourcing”, that allows the users to add a schedule while they are traveling in a bus. The users have to select “Start Crowdsourcing” so that the data is collected as background service. The “Start Crowdsourcing” screen is shown in Fig.2.

This background service listens to the android accelerometer application and when the accelerometer reading is greater than a threshold value the GPS is turned on and speed is monitored. If the speed is close to zero, it is taken as a stop and at that time the GPS is turned on and the coordinates of the stop location and time are added to the local database on the phone.

There is another background service which listens to the internet connection availability. Whenever the connection becomes available the data along with the user ID is sent to the server.

Fig.2. "Start Crowdsourcing" Screen

Fig.3. "Find Your Bus" Screen

2) Display the Bus Schedule as per user query: BusTimer application users can query the bus schedules by providing the source and destination places using the option "Find your Bus", whose screen is provided in Fig.3.

To make the querying easier, an auto complete feature is implemented. All the schedules that have bus stops in both starting and destination are shown with the ones after the requested time highlighted.

B. Back-End

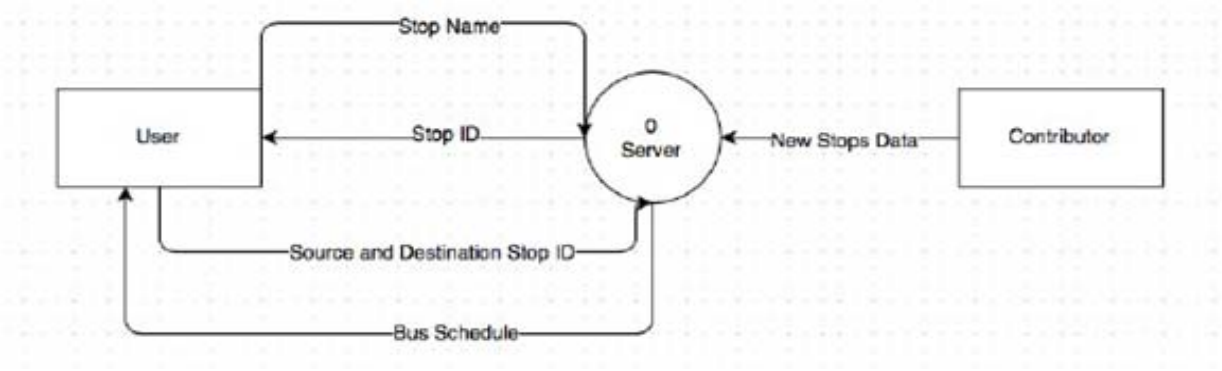
The system back-end has two main components

1) Web Server: It handles all the interactions with the user. Whenever a new data about bus stop is submitted, the webserver pre-processes the data and stores it in Amazon SQS (Simple Queue Service). Also it is the web server that responds to the user bus schedule query by providing the results.

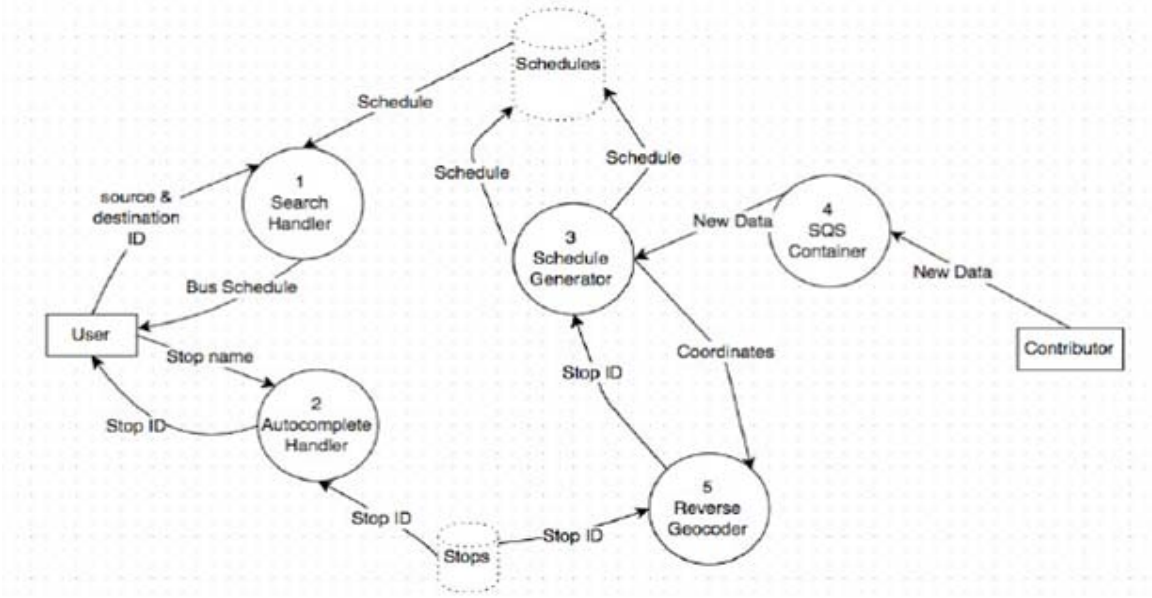
2) Worker Server: The Worker runs the bus scheduling algorithm. It has the following functionalities:-

- *Autocomplete handling*:- As the user starts entering the source and destination in the App, the auto-complete functionality searches for stops having the similar name and returns a list of stops to the user. This enables the user to enter data faster as well as to recollect spellings of stops they are not familiar with.
- *Generating User IDs*:-Unique user ID's are provided to each registered user when he/she app and uses the app for the first time. This is generated by the worker server.
- *Reverse Geocoding*: - The android app will collect GPS coordinates of the bus stops identified by the accelerometer. But in order to resolve these coordinates to actual place names of bus stops, we use Google Map's Reverse Geocoding API. Once supplied with a latitude and longitude, this API will return the bus stop near that area. However, Google restricts the total number of API requests per day to 2500. Hence all the coordinates received cannot be sent to API directly. Hence we cache the results. Once the places are resolved, we store the place details and coordinates in the stops table. Hence for the next time, we do not have to search GMap API instead we can simply query the stops table.

  There is another setback here. That is, the table will contain an accurate latitude and longitude of the stop. But the data received from the App need not be as accurate as the accuracy depends on various factors. Hence, we assume the collected coordinates as a particular stop, if the latitude and longitude come within a threshold of the stop (approx. 300 m).



(a) Level 0 DFD



(b) Level 1 DFD

Fig.4. Data Flow Diagram for BusTimer

- *Generating Bus schedule between the source and destination*: - The actual schedule, which is provided to the user by web server, is generated by this worker server through table look up. The bus schedule generation algorithm is mentioned below.

The level 0 as well as level 1 Data Flow Diagram for the proposed system is shown in Fig. 4.

IV. Bus Schedule Generation Algorithm

Each of new data set received through crowdsourcing is sent to the schedule generation algorithm. The algorithm works as follows

1) Step 1: Iterate through each stop in the new data and reverse geocode it. At the same time, also check if the resultant stop is part of any of the existing schedules. This will be stored in a schedule counter which contains schedule IDs and the stops that come under each schedule.

2) Step 2: In order for identifying whether the newly received data is a part of an already existing schedule, we have to obtain the count of the prominent schedules. A newly received schedule is counted as a prominent schedule if it contains 2 or more existing stops.

3) Step 3: Based on the count of the prominent schedules identified the following procedure is done.

- If there is no prominent schedule: - Construct a new schedule out of the obtained data.
- If there is exactly one prominent schedule: - This means that the majority of stops in the dataset is already a part of an existing schedule. But we will be adding this to the prominent schedule only if the first and last stop in the new dataset is a part of the prominent schedule or the first and last stop in the prominent schedule is a part of the new data set. Else, we will create a new schedule using the obtained data set.

- If there is more than one prominent schedule: - This means that the obtained data set is a part of two or more schedules. Then we could merge the two schedules, but merging is done only if the top part of one schedule is the bottom part of other schedule. Otherwise a new schedule is created using the obtained data set.

Step 4 - Adding a new stop to an existing schedule: - If the stop is not already in the schedule, then it has to be added at a suitable location. To do so first we plot the stops and time along a circle with the time moving in clockwise direction. Now for each pair of stop in the schedule, we check whether the new stop comes in between the pair. If it does fall in between, we add the stop there. And if it is not in between any of the stops, it has to come either first or the last position of the schedule. We can decide this by calculating the distance from the new stop to the first and last stops. Whichever is smaller, we insert the new stop at that point.

V. Implementation and Testing

The front end of BusTimer application is installed in an android based mobile phone that has accelerometer sensor and GPS and GPRS facility.

The back-end of our system is hosted in Amazon Web Service (AWS). AWS is Amazon's cloud hosting service. Since our app is hosted on AWS, it will scale automatically as the traffic increases. The data is stored in Amazon DynamoDB. Also, an Amazon SQS is used for temporary storage.

Back-end Web Server is a python based HTTP server built on top of the flask framework hosted on AWS; that is run on a tc2.micro EC 2 instance.

Back-end Worker server is also a python based HTTP server built on top of the flask framework. Hosted on AWS, that is run on a tc2.micro EC2 instance. It has an SQS daemon running on it, which constantly listens to the Queue. When a new data arrives in the SQS, the daemon will send its data to the flask web application running on the instance. The web application runs the scheduling algorithm and the results are updated in the schedules table in the DynamoDB.

In order for crowdsourcing, we needed users to travel by bus with the BusTimer application installed android phones. 25 people volunteered to test the application for us. The people were chosen so that the routes they travel overlap each other since we need a minimum no of people to give the similar data for the algorithm to transform them into schedules. Fig.5. shows the *users* table.

User_id	User_key
8aeb59b683948df29bf5d06c1c7af7183ab9292e00563352dea67bc66fb036fb	6ef5e2c62e0393b54be9154c265dfdec7fc89b39446797ee95856be
9b7eec0c175dea647cfb2ece805b1373b76a17c253611766280f0bd8403ed3a5	7b6c4a69ac2b1524210f2a9091fdee74f4b97fdb0efac394998746a
45abaf662ee5060b55297a17c585a87f341001162dead93f4043610f399c33e9	b75e72201cdb875feeb43b0a099b9961fc9e2e66308e930c79c288e
dccc04c95f29cc62d78414108ba8c3d1c1b46634eaa989f9d432ec9238a51e42	4c1efdaf104e815c3d05c21b8e64d8450eeaf2542f1c93ec4ac494e
0d2c03a4415acda63bebaeb92af5bda8705c9b7e301119d3fabe83a25c727fe3	43f525ab35342f81bf5c28c294c4122f230699dc1c8ace63119b8dd
ec4d422f4984e3f6f1bff9003fac3cb1ea90dee4fcdf780339223c15a35eb3b3	398160ab10e0bf016910d073c692d01a7365e8d5e656443258edce6
22b38060f39566d9c02f877c96e31de2094d08	8b7e35181c87a7aadfb5ba8cfd8f47

Fig.5. Users Table

Stop_id	latitude	Longitude
74868daa44cbca08e9e309ea8a5d7ae2cf956d45	9.09231	76.49408
f7f37f3031950add1a683aa004c7aed02e0a5b8d	9.10067	76.52202
1343c20187fd3281e81253e9e23acc4450d493d7	9.08552	76.52924
3ae07536fd7706357a246b3d1e75a5855baf23a4	9.07724	76.53196
b178387cc0bb34c650b803d8f9263f19d0f6f009	9.0522	76.53634
d90a41f75f1f8f92ebbb8b74f9e310c8af7aa4b7	9.04921	76.53601
ae540f6d43b09c5019fd7743745ed80100397230	9.04504	76.53571

Fig.6. Stops Location Table

We managed to collect 80 data sets through crowdsourcing and the corresponding locations are added to stop location table (Fig.6), and added to the *stops* table, which is shown in Fig.7.

The data is then passed into the scheduling algorithm to generate schedules. During data collection, our application successfully distinguished the bus stops based on accelerometer readings and activated the GPS to send the location coordinates to the server.

In our testing, we were able to form seven schedules using the received datasets out of eight actual schedules. The absence of one schedule is due to the low no of contributors, because of which the scheduling algorithm cannot confirm it as a schedule hence it is not added to the schedules table till further datasets belonging to the same schedule are received. The *schedule* table is given in Fig.8. From the seven schedules generated, two of them belong to the same route which is detected by the scheduling algorithm and they are merged together.

The bus schedule displayed to a user who queried with the "Find your Bus" option shown in Fig.3. is shown in Fig.9.

VI. Conclusion

This paper discusses a low-cost energy-efficient system for generating bus schedules using Accelerometer, GPRS, and GPS on GSM network through crowdsourcing. The combination of both the technologies i.e. GPS and GPRS provides a constant, reliable tracking system. The cost of the overall system has been reduced by two facts one is using the mobile phone based GPS instead of a separate GPS device and another is by using GPRS instead SMS. The battery is conserved by activating the GPS only when the accelerometer identifies a stop thereby making the system energy-efficient as well. Thus this projects presents an application to provide a bus schedule, even for short distant journey.

ID	Name	Level 1	Level2	country	Threshold
74868daa44cbca08e9e309ea8a5d7ae2cf956d45	Vallikkavu	Kollam	Kerala	India	6
f7f37f3031950add1a683aa004c7aed02e0a5b8d	Vavvakavu	Kollam	Kerala	India	4
1343c20187fd3281e81253e9e23acc4450d493d7	Puthentheruv	Kollam	Kerala	India	5
3ae07536fd7706357a246b3d1e75a5855baf23a4	Puthiyakavu	Kollam	Kerala	India	4
b178387cc0bb34c650b803d8f9263f19d0f6f009	Karunagapally	Kollam	Kerala	India	4
22b38060f39566d9c02f877c96e31de2094d08d7	Lalaaji Junction	Kollam	Kerala	India	5
74cacdad246c273a66342ea3215d27c45862d0b3	Karottu junction	Kollam	Kerala	India	6
db9dcf11904c3fa5430d6d7a1cfc32dab822d274	Kuttamukku	Kollam	Kerala	India	5
e9aa2ccd136291c7b17c6465147ac00212a84802	Joint Junction	Kollam	Kerala	India	5

Fig.7. Stops Table

id	Schedule_id	Stop_id	time	contribu ors
	f7145d99bf892fe400049a533246fdc698178b50	74868daa44cbca08e9e309ea8a5d7ae2cf956d45	355985	8
	3b13e86ac359368336c984de4ac963f32c71bfac	f7f37f3031950add1a683aa004c7aed02e0a5b8d	97556	8
	c168d403055e9a94145ecc77abed2e5598b8998f	1343c20187fd3281e81253e9e23acc4450d493d7	355985	8
	1fb0e34032ea6eac03989b1b7ce7750d6041a0ad	b178387cc0bb34c650b803d8f9263f19d0f6f009	425842	3
	f2598de4cb8b0c8510fac3e8b6fea829801df9a2	3ae07536fd7706357a246b3d1e75a5855baf23a4	552967	24
	910cb87ae7436c119aa96bbbafc970fd8965e564	ae540f6d43b09c5019fd7743745ed80100397230	478258	17
	97762a3a9b72a91ca8f1a10dc919a7f53e836859	c5d5f4e99d3a9c26fdb60f4cd8cbae7b76338820	212615	12

Fig.8. Schedule Table

Fig.9. Bus Schedule Query Result Screen

REFERENCES

[1] Binu P.K, Viswaraj P.S, "Android based application for efficient carpooling with user tracking facility", 2016 IEEE International Conference on Computational Intelligence and Computing Research, ICCIC 2016.

[2] J. Amudha, Hitha Nandakumar,S. Madhura, M. Parinitha Reddy, Nagabhairava Kavitha, "An android-based mobile eye gaze point estimation system for studying the visual perception in children with Autism", 1st International Conference on Computational Intelligence in Data Mining, ICCIDM 2014.

[3] C. Vishal, V., R. Shivnesh, Kumar, V. Romil, Anirudh, M., P. Bhagavathi Sivakumar, Velayutham, C. S., Suresh, L. P., and Panigrahi, B. K., "A crowdsourcing-based platform for better governance", Proceedings of the International Conference on Soft Computing Systems, Advances in Intelligent Systems and Computing, , vol. 397, pp. 519-527, 2016.

[4] Peng X., Gu J, Tan T.H, Sun J, Yu Y, Nuseibeh B, Zhao W, "CrowdService: Serving the individuals through mobile crowdsourcing and service composition", proceedings of 31st IEEE/ACM International Conference on Automated Software Engineering, pp. 214-219, 2016.

[5] Hasan, K. S., Rahman, M., Haque, L. A., Rahman M. A., Rahman, T. and Rasheed, M. M., (2009), Cost E□ective GPS-GPRS Based Object Tracking System, Proceedings of International Multiconference of Engineers and Computer Scientists, March 2009, Vol-I.

[6] Fengli Zhang, Xinggao He, Bo Xu, and Minglin Deng, Integrating Moving Objects Location Data with GIS-Based Web Environment.Communications, Circuits and Systems and West Sino Expositions, IEEE 2002 International Conference on, 2002

[7] Yuan, G., Zhang, Z. and Wei Shang Guan, (2008), Research and Design of GIS in Vehicle Monitoring System , IEEE International Conference on Internet Computing in Science and Engineering.

[8] Aloquili, O., Elbanna, A. and Al-Azizi, A., Automatic Vehicle Location Tracking System Based on GIS Environment, IET Software, 2009, 3.4, pp. 255-263.

[9] Al-Bayari and Sadoun, O. B., (2005), New Centralized Automatic Vehicle Location Communications Software System under GIS Environment, International Journal of Communication Systems, 18.9, pp. 833-846.